

*Focal Point Foundations*

```
142 // ...
143 // ...
144 // ...
145
146 HOURS = 360000
147 MINUTES = 60000
148 inclusionThresholdWarningDefault = false
149 OVERRIDE_LOCK = { 'headline': 'combined warnings', 'js':
150 HazardTypes = [
151   'af', { 'headline': 'ADHOC ADVISORY',
152         'override lock': OVERRIDE_LOCK,
153         'combined warnings': true,
154         'allowFull': true,
155         'allowFullChange': true,
156         'allowFullChange': true,
157         'expirationTime': 60, 300,
158         'expirationTime': 60 * MINUTES,
159         'expirationTime': 60 * MINUTES,
```

*Click “Next” to begin the presentation...*

[No audio for this slide]

**Warning Decision Training  
Division**

Presenter:

**Eric Jacobsen**

Course Contacts:

**Eric Jacobsen**

[eric.p.jacobsen@noaa.gov](mailto:eric.p.jacobsen@noaa.gov)



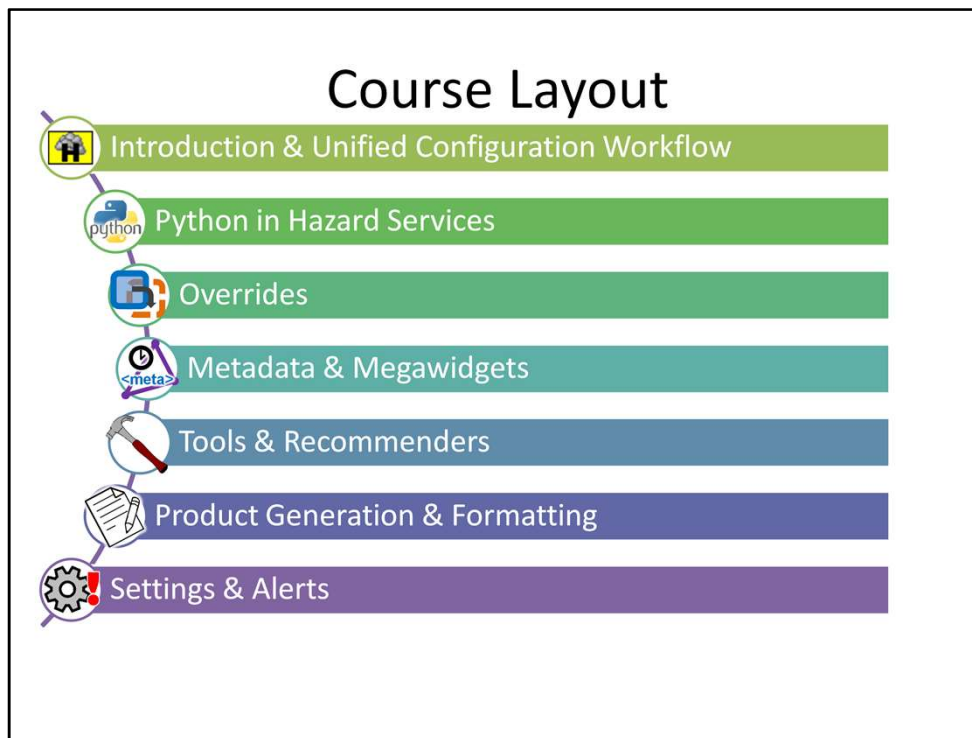
**Hazard Services,  
Focal Point Foundations Course**

## **Course Introduction & Unified Configuration Workflow**

Welcome to the Hazard Services Foundations training course for Focal Points.

This module is the first in the course, and introduces focal points to Hazard Services' Unified Configuration Workflow.

My name is Eric Jacobsen, with the Warning Decision Training Division. If you have questions about this course, or technical problems, please use the contact information listed on this slide.



This course is a collection of seven independent modules, which we recommend taking in the following order.

This module on the unified configuration workflow introduces many of the components studied in later modules. An overview of key Python concepts is given next. Overrides discusses strategies for altering Hazard Services files. Metadata is covered next, and its related topic of Megawidgets... followed by Tools & Recommenders... and then, a basic overview of the complex topic of Product Generation and Formatting. We also discuss some of the key settings and alerts that focal points must be aware of.

Each of these modules has its own objectives and quizzes, and can be revisited at any point.

## Unified Workflow: Objectives

After completing this module, you will be able to identify:

- The **programming language** used for configuration
- The **extent of customization** possible
- The **two** main **functional groups** for configurations
- The functional groups supported by:
  - Types & Categories, Metadata, Recommenders
  - Utilities, Generators, and Formatters
- The **key step** separating hazard creation and product generation
- The role of the **registry**
- Local vs. remote **storage of hazards and products**
- The **responsibility for maintenance** of Hazard Services and legacy software
- How hazard services files need to be shared for **service backup**

These are the objectives for this module. Please take a moment to review them, then when you're done, click next to proceed with the module.

## Hazard Services' Unified Workflow

- Unified Configuration Workflow
  - Consistency between products
  - Flexibility to grow
- One configuration language (Python) with shared code across hazards
- *Extremely* customizable
- Separation between:
  - managing hazard information
  - producing products



Just as the Hazard Services user training demonstrated the consolidation of many legacy tools into a single workflow, the *configuration* of these different products has likewise been unified in Hazard Services. This unification achieves much needed consistency between the setup of products previously handled by the legacy apps: RiverPro, GHG, and WarnGen.

In addition, Hazard Services' underlying framework delivers the added benefit of flexibility for changing the way information is managed and leads to product creation, especially relevant as Weather Service strategies evolve.

The primary features of Hazard Services which give it a capacity to extend and envelop many legacy and future capabilities are: A common language for configuration (chiefly python, with very few exceptions) with shared code for all hazards; that it is extremely customizable (almost any component can be customized); and a distinction between managing hazard information and producing products. Let's understand how central this last concept is to the Hazard Services workflow...

# Separation of Hazard and Product

Time Range

Start: 01-Jul-2015 23:00

End: 02-Jul-2015 07:00

Event ID	Hazard Ty	Site ID	Status	Sta
619	FA.A	LWX	PENDING / PROPOSED	23:2
607	FLY	LWX	ISSUED	06:5
609	FLY			06:5

hazard

- Focus on hazard
  - ✓ Area
  - ✓ Time
  - ✓ Type
- Share metadata

Product Editor

FFW

404 - LWX

VAC043-047-061-107-113-153-157-187-WVC037-272315-  
/O.NEW.KLWX.FF.W.0020.180527T2017Z-180527T2315Z/  
/00000.0.ER.000000T0000Z.000000T0000Z.000000T0000Z.OO/

Basis Time: (\* required field)

At 417 PM EDT,

product

producing heavy rain across the warned  
fallen. Flash flooding is ongoing or  
ected to begin shortly.

- Multi-format capability
- Shared data  
(supports consistency)

A key pillar of Hazard Service’s flexibility, and one which infused into the user and configuration experiences, is how it segregates the management of hazard information from product generation.

This flexibility is embodied in two core capabilities: Hazards can be identified, shared and maintained without regard to a product; and, any number of product formats can be generated simultaneously from shared data and attributes.

# The Two Main Dictionaries

- **One per hazard event**
- **Includes:**
  - Type, Time, Area
  - Metadata
- **Sharable**
- Configuration workflow mirrors same distinction
- Actual Python code centers around building, passing, and referencing these dictionaries

- **One per product**
- **Product information and phrases**
- **High-level product structure**

Supporting this separation behind the scenes, for a given hazard-to-product evolution, information is centralized in two main storehouses, which for practical purposes, we simply refer to as “dictionaries” after the Python variable type.

The Hazard dictionary stores attributes which characterize a hazard. This includes the time, location, and type of hazard, and descriptive metadata, which will be covered in more detail shortly. The product dictionary, meanwhile, is an all-encompassing collection of key product information and phrases, structured to follow the rules that product dissemination requires.

As a quick clarification, although hazard events are not strictly stored as python dictionaries, most of the usable data in Hazard Services for both hazards and products is ultimately presented in dictionary form to users performing advanced configuration on product output, and so we’ll be forgiven for making this simplification.

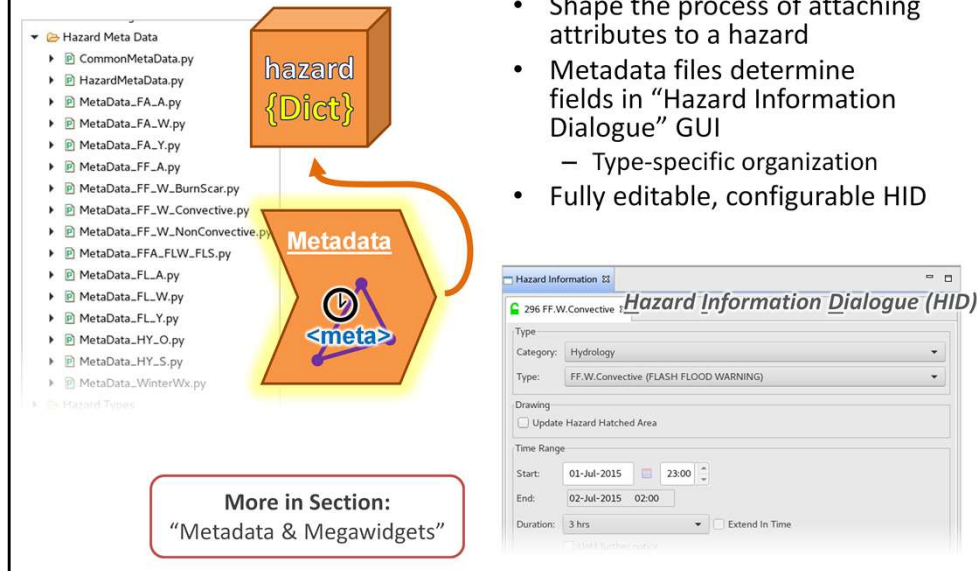
The relationship between hazard and product is extremely context-dependent in Hazard Services, but the process of transforming a hazard to a product starts with a user clicking “Preview” in the Hazard Information Dialogue.

This very abstract representation of the Hazard Services product flow underlines the overall break-down which focal-points must expect in their configuration duties. In fact, neither dictionary maps to an actual, viewable file, but most of Hazard Service’s

configuration files revolve around them, and the two dictionaries are crucial variables throughout the backend code.



# Hazard Workflow: Metadata

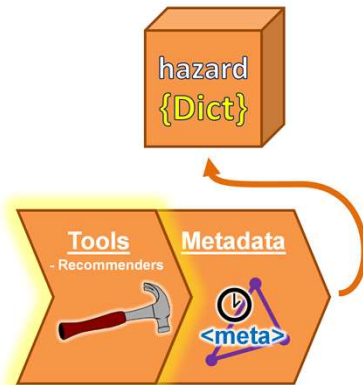


What are some of the files related to the hazard dictionary?

To start, Hazard metadata, encompassing all of the auto- or user-selected attributes for a hazard type, are stored in the hazard dictionary for that specific hazard event. Metadata configuration files, shown here grouped in their own localization directory, principally define which fields appear on the "Hazard Information Dialogue" window, or "HID" in short, for a given hazard type, such as the example shown on screen for Convective Flash Flood Warnings. For each hazard, the HID guides the selection of attributes that characterizes it, in turn directly dictating what will be stored in the hazard dictionary. Given the often specialized set of attributes needed to characterize different hazard types, a separate metadata file will typically exist for configuring each.

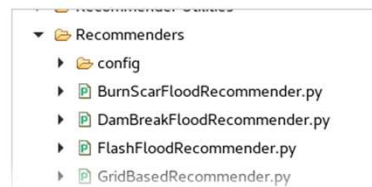
Metadata files are fully editable, and focal points have a high level of control over the HID that symbolizes this stage of the hazard event workflow, simply through configuring each metadata file. This crucial component is covered more thoroughly in a dedicated section.

## Hazard Workflow: Tools



More in Section:  
“Tools & Recommenders”

- Act on or help produce hazard events
- Python scripts
  - data access capabilities
- Recommenders:
  - Tools specifically focused on event suggestion and metadata
- Fully editable



Another core component of hazard creation is Tools. In general, Tools are powerful helpers which act on or help produce hazard events. All tools are python scripts whose starting focus is on hazard data, but may serve a range of purposes spanning multiple functional areas of Hazard Services. Most also leverage a powerful data access framework, to be discussed later.

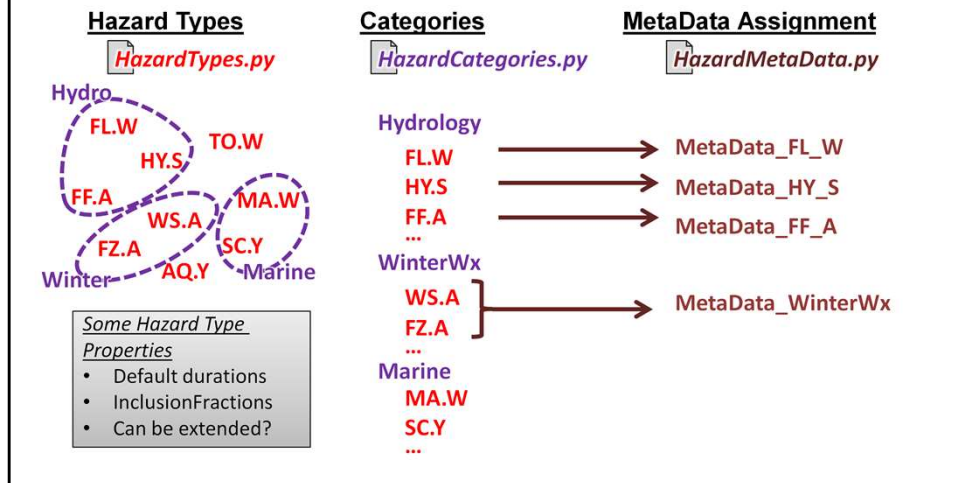
“Tools,” in fact, is a broad reference which also encapsulates an important subtype, “Recommenders.” Recommenders are much more specifically designed to present forecasters with hazard candidates and kick off the population of hazard metadata.

In our conceptual outline of the hazard workflow, tools, and in particular recommenders, *usually* lead to metadata, and in this way ultimately help contribute to the dictionary for a given hazard. Some exceptional tools may, again, reach beyond this scope, but most almost always start with managing hazard events, and thus primarily find their home on this side of Hazard Services’ functional division between hazards and products.

As a crucial component in the hazard event workflow, the python files behind tools have their own folder in the localization, shown on screen. These files, and their associated behavior, are completely editable. Tools and recommenders are a rich area for configuration, and therefore a separate section is dedicated to them later in this training.

# Simple Hazard Management

- Central files declare, organize, and link hazards



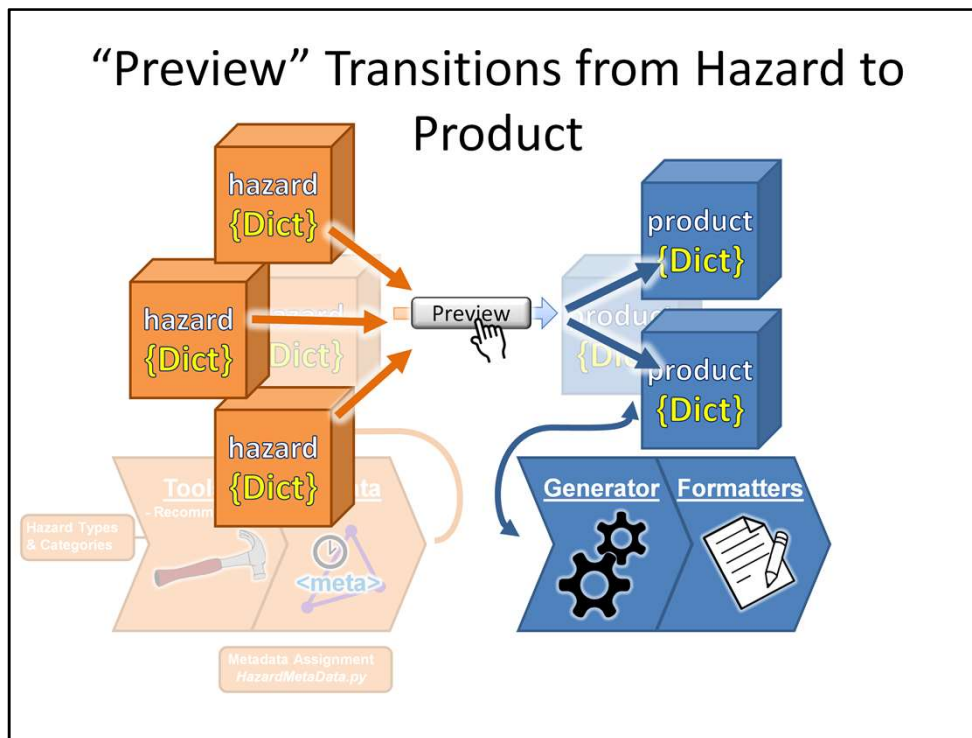
Because hazard management is such a fundamental activity in Hazard Services, that part of the workflow is explained in more detail here.

Localization files for hazard and metadata specification are among the most straightforward to view and edit in Hazard Services, which is fortunate given their importance. In fact, just three central files govern the definition, categorization, and mapping of hazards to metadata.

To be of any use, a hazard must first and foremost be included in an all-encompassing dictionary of possible hazard types, aptly named “HazardTypes.py”. This file contains fundamental properties for hazards, and while many of these are fixed by directive, such as whether it can be extended in time or area, other properties like default duration can be edited here and ultimately help populate hazard information.

Because the list of hazards is sizeable and will only grow as other capabilities are added, a categorization file (HazardCategories.py) is provided to organize these into groups. This relatively straightforward file includes a list of default categories and the hazard types belonging to them, and, as with almost everything in Hazard Services, it is completely editable, allowing groups to be added, removed, or changed. One of the main purposes of this grouping is organize and simplify the user experience to help focus on particular tasks.

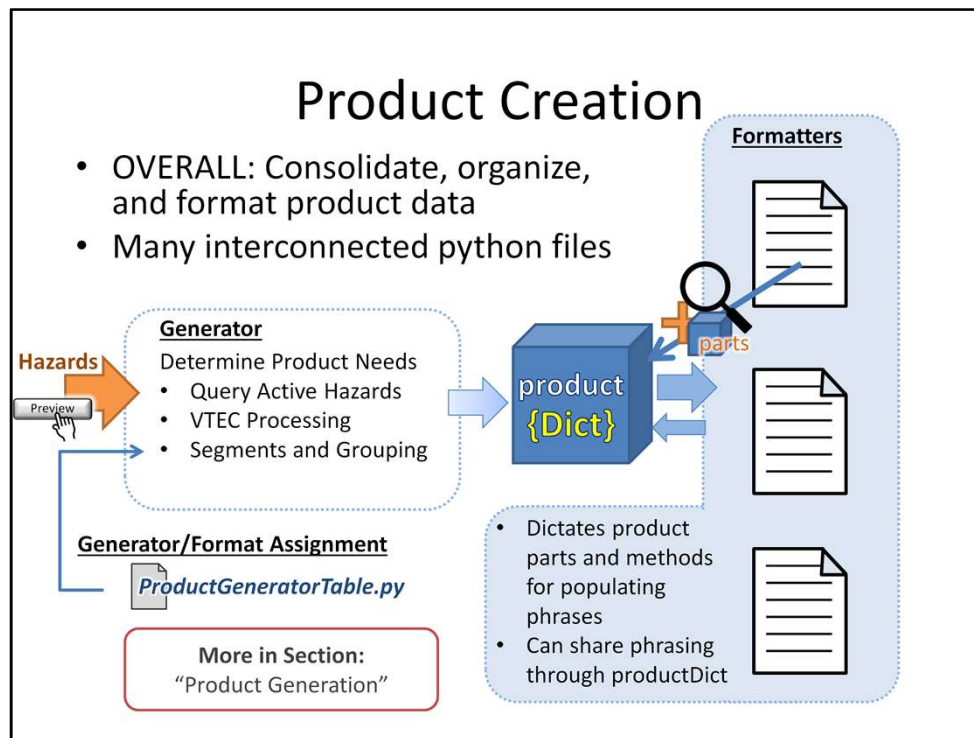
Each hazard type is associated with a metadata file, except in the unlikely case where no attributes are needed. This mapping is handled with the HazardMetaData.py file, shown here. Due to the unique nature of most hazards and the specificity required of their metadata configuration, many hazards are mapped to a metadata file of their very own. However, multiple hazards may share a single metadata file if their pool of potential attributes is nearly identical.



All the hazard-related configurations support the job of identifying and characterizing, through the HID, each hazard that might be presented to the forecaster.

The transition from hazard to product is embodied in the “Preview” button of the HID. Symbolically, this action leaves the world of hazard configuration and shifts to the product generation side of Hazard Services.

Ultimately one or more hazard dictionaries translate to one or more product dictionaries, in a complex process which in actuality is a combined effort of the product generators, formatters, and a variety of supporting utilities. This product workflow is the topic of the next slides.



Hazard Services' product side can be summarized with the overall purpose of consolidating and organizing information related to products, and formatting the desired output. This deceptively simple summary encapsulates a massive amount of logic, utilities, and interdependent files.

Before any generation begins, the authoritative Product Generator Table determines how ALL hazards are mapped to a generator, and to various previewed or issued format options.

The product generator consolidates any needed information, which in addition to metadata/attributes from the hazard dictionary includes information about active products, and initializes a dictionary for each product it determines to be needed. It organizes this dictionary with placeholders for the high-level product structure.

Next, the chosen formatters (with Hazard Services enabling multiple to act on a product) work directly with data from the product dictionary, and construct the low-level product parts which are then translated into text and phrases. These phrases and parts can also be contributed back to the product dictionary, which allows for shared phrases and maintaining consistency across formats.

Again, product generation is a complex process consisting of many interdependent files. Some of these files, and important features of this product generation in general, will

be covered in more detail later in a dedicated section.

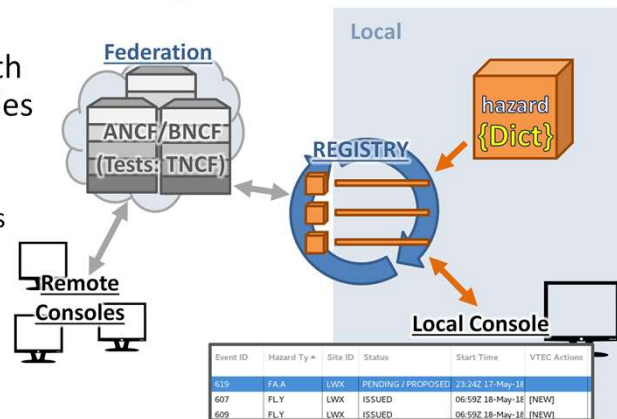
## Registry Syncs Hazards

- Where does **Hazard** data go?

- Synchronized locally **and** with remote consoles via Registry

- “Backup” or “Visible” sites

- Not Archived



Although we’ve covered the big picture of both hazard and product configuration, one other central aspect to Hazard Service’s management of data remains. This has to do with where hazard and product data are shared or stored.

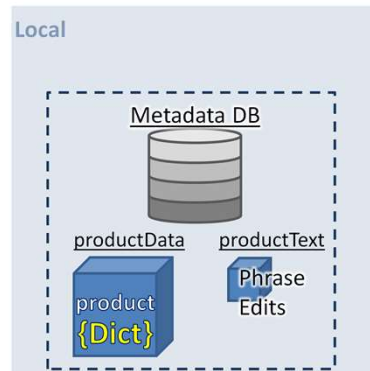
Hazard event data, recall, is stored in a dictionary of its own for each event. These event-specific dictionaries are managed by a background service called the “Registry.” This registry serves two important roles: First, this service synchronizes information about all hazards and updates this information on other local consoles; in addition, the registry passes local hazard event information to a federation which manages all national hazards. This two way exchange with the federation, which operates on ANCF/BNCF, or TNCF during tests, allows local hazard events to be visible on remote consoles, and for hazards from other sites to be retrieved for display on local machines. Specifically, events are retrieved for sites which your office has configured as “backup” or “visible” sites in Hazard Services. This exchange is limited only to hazard event information, not products, and exemplifies the very distinct treatment of hazard data from their product counterparts in Hazard Service’s workflow.

Although this distribution of hazard event data supports live, situational awareness... it should be noted that they and their backend registry objects are NOT included in routine or case archiving. From an archive dataset perspective, the hazard event dictionaries are simply a means to an end, that end being product generation, and thus are not retained indefinitely.



## Product Data Stored Locally

- Where does **Product** data go?
- Dictionary elements stay local
  - **NOT** synced via Registry
  - Metadata DB:
    - **productData**: full product dictionaries
    - **productText**: phrase edits
      - **Previous Text** checkbox in Product Editor window
  - Not easily viewable
  - Not Archived
- Text products distributed as before



Basis Text: (\* required field) ☒ Previous Text

trained weather spotters reported a cluster of thunderstorms producing heavy rain across the warned area. Between 2 and 3 inches of rain have fallen. Flash flooding is ongoing or expected to begin shortly.

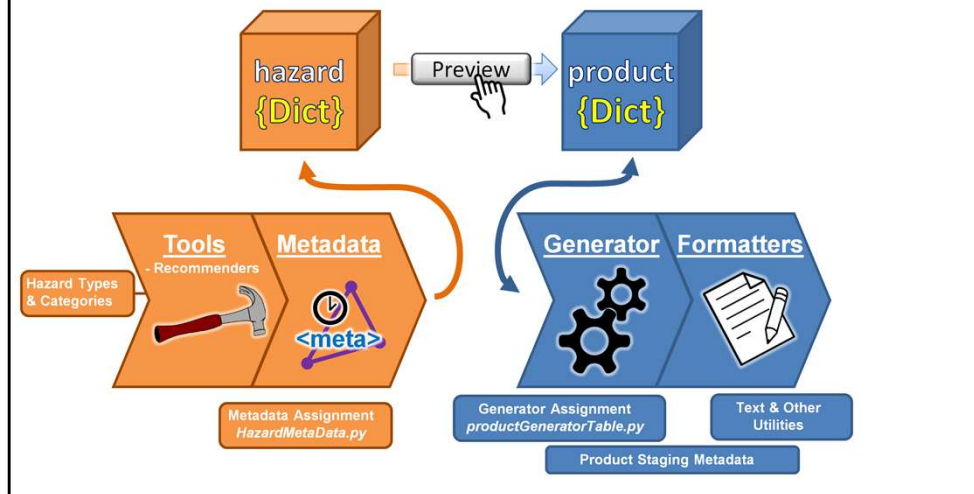
Persistence of the product dictionary is quite different. In contrast to hazards, the product dictionary is *NOT* synchronized through the registry and stays within the local office.

In the local metadata database, Product dictionary elements are stored in a “productData” table, and a “productText” table exists solely to track user edits which were made to product parts through the product editor. This latter table in fact stores the edits that, if available, are recalled in the product editor when the “previous text” option is checked for a particular field.

In general the product dictionary is not easily reviewed, and has no corresponding tool like the “event details viewer” for hazards. In addition, neither of these tables are currently archived... product dictionaries are basically internal data that help Hazard Services do its job.

Although the product dictionary itself is not shared with remote sites, the standard text products which are their end result are distributed as before.

# The Hazard Services Workflow



One key aim of this section was to provide a birds eye view of the entire Hazard Services workflow from hazard management to product generation. The entire workflow is depicted on this slide, emphasizing the core components which interact with the two main divisions: hazard, and product. Separate modules in the rest of this training dive into these topics in greater detail.

The remaining slides in this section transition from this walkthrough of the components, to emphasizing some of the key overall characteristics of Hazard Services.

## Extensive Configurability

- Including:
  - GUIs and dialogues (e.g. Hazard Information Dialogue)
  - Tools/Recommenders
  - Hazard Definitions, Metadata
  - Product Generators, Formatters
  - and more
- Python, extensive editing
  - **CAUTION:** Configure precisely to minimize negative impacts

Details **EXAMPLE HID (FF.W)**

IBW type:

- ☒ General
- ☐ Considerable Flash Flooding
- ☐ Include Flash Flood Emergency / Catastrophic Flooding

Enter location

Immediate Cause: ER (Excessive Rainfall) Immediate Cause

Source:

- ☒ Doppler Radar indicated
- ☐ Doppler Radar and automated gauges
- ☐ Trained spotters reported
- ☐ Public reported
- ☐ Local law enforcement reported
- ☐ Emergency management reported
- ☐ Satellite estimates
- ☐ Satellite estimates and gauge reports
- ☐ Gauge reports

Source

Event type: Thunderstorm(s) Event Type

☐ Flash flooding occurring Flash Flood Occurring

Rain so far: Unknown Rain Amount

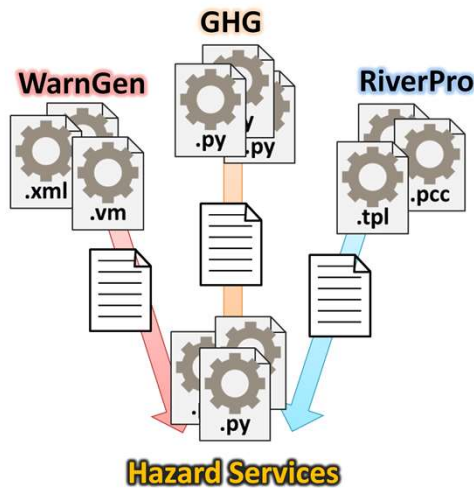
```
def execute(self):
    self.init()
    if self.hasMetadata():
        metaData = self.getIBW_Type(),
        self.getImmediateCause(),
        self.getSource(),
        self.getEventType(),
        self.getFlashFloodOccurring(),
        self.getRainAmt(),
        self.getRainRate(),
        self.getAdditionalInfo(),
        self.getFloodLocation(),
        self.getLocationsAffected(),
        self.getAdditionalLocations(),
        self.getCTAs(),
    ]
    self.getAdditionalInfo(),
```

As the focal point navigates each component to manage the way Hazard Services operates, a central characteristic will be the openness of virtually all of its code.

In Hazard Services, almost limitless customization can be applied to: The front-end GUIs and dialogues, particularly including the Hazard Information Dialogue; the tools and recommenders and what each does; the back-end set of hazards definitions and Metadata; the back-end processes of product generation and formatting; and much more. Fundamentally built on very dynamic python files, possible configurations range from tweaking a small interface component in the HID, to completely changing the way a product formatter works.

As more and more legacy capabilities are rolled into Hazard Services, this dynamic nature will allow the underlying single workflow to have a powerful degree of flexibility. But with such freedom, focal points should be particularly cautious about best practices in their configuration, making precise edits to minimize any potential negative impacts, even while the openness of python allows any edit to be made.

## Interoperability with Legacy Apps



- Hazard Services eventually replaces capabilities of legacy software
- Interoperability:
  - Parallel accessibility of products with legacy apps
  - REQUIRES parallel configurations management
- Functional Interoperability:
  - **Maintain changes in BOTH legacy apps AND Hazard Services**



**Legacy support  
will be retired**

Hazard Services will gradually take the place of legacy software in producing operational products, starting with hydro, then winter, marine, and so on. Rather than immediately retiring the legacy applications, however, Hazard Services was implemented in a way that allows temporary parallel use of it and legacy applications, formally referred to as “interoperability.” This means an ability to issue products in one context that are accessible in another.

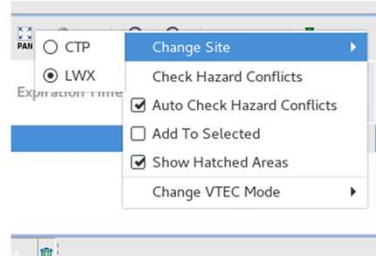
However, it does require that focal points maintain parallel systems so that their operation is consistent with each other. Many configurations are not shared between legacy applications and Hazard Services, thanks in part to the different file formats used for configuration by each. For example, for the geospatial config thresholds in WarnGen, Hazard Services has its own independent files which are not synchronized.

In general, focal points should expect to front-load some time synchronizing Hazard Services’ baseline performance with essential customizations in their own legacy applications. This will not only support functional interoperability while the capability exists, but is a fundamental task in migrating to Hazard Services.

To re-emphasize, however, while interoperability with legacy applications exists initially to support the transition to Hazard Services, the corresponding legacy software product generation *will be* discontinued sometime after full deployment.

# Backup Operations

- Hazard Services backup support:
  - Easy transition to backup (see “Settings & Alerts” section)
  - Registry-based hazard sharing strengthens data flow
- Maintaining backup localizations:
  - **Regular update of Hazard Services directories** within backup site localization is crucial



Related to maintaining seamless, local configurations, focal points must also be conscious of the need to maintain localizations for backup sites.

Hazard Services' interface and registry-based hazard sharing supports convenient transfer to backup operations (covered more in the “Settings and Alerts” section), but, like many legacy applications, it requires timely retrieval of those site's localization files to work properly. Beyond initial steps to set which sites can be backed up, regular updates of the Hazard Services localization directory should be folded in with existing transfers of a backup sites most recent utility tree.

# Hazard Services Support

- GSD Hazard Services Focal Point Guide

- [Google Docs Link](#)

- Individual Support

- **Software Crashes:**

- NCF

- **TEST-PHASE Support:**

- [NWS.HazardServicesTeam.Staff@noaa.gov](mailto:NWS.HazardServicesTeam.Staff@noaa.gov)

- NWS Chat room:  
Hazard\_services\_support

- **Post-deployment Support:**

- awips2dev listserv



Hazard Services Focal Point User's Guide

Getting started -- how to achieve minimal functionality.

A. Customizing StartUpConfig.py.

B. Define backup sites if needed.

C. Synchronize Hazard Services inclusion percentages.

D. Convert warnGen information about predefined useable by Hazard Services.

Chapter 1: Customizations

Audience: Focal Points

1.0 Localization -- How to set up Hazard Service

1.0.0 A note about Region level override.

1.0.1 Recommended Steps

1.0.2 Configuring Hazard Services for pre impact areas.

1.0.2.1 What parseWarnGenTempl

1.0.2.2 How Hazard Services uses impact areas.


1.0.2.3 Situation where warnGen w predefined impact areas.

Finally, although the upcoming modules throughout this training will explore many aspects of Hazard Services configuration, focal points facing the breadth of topics and configuration needs will certainly need additional support and documentation. A few key support resources are highlighted here.

The Hazard Services focal point guide, maintained by developers at the Global Systems Division of ESRL, is an in-depth document covering many of the upcoming topics, and will be referenced heavily.

For individual support, the following channels are available depending on the situation. Generally, for software errors or crashes, contact the NCF. For all other questions including configuration, during site tests and BEFORE official deployment, focal points can either contact [NWS.HazardServicesTeam.Staff@noaa.gov](mailto:NWS.HazardServicesTeam.Staff@noaa.gov), or use the a Hazard\_services\_support room available in NWChat. After deployment, support will transition to the awips2dev email list, which can be used for posing questions to the community, whose responses can benefit all subscribers.

## Unified Configuration Takeaways

- One configuration workflow
  - Unifies formerly separate legacy apps
  - Almost anything is editable with Python
- Configurations fall into two functional groups
  - Hazards:
    - Types & Categories, Metadata, Recommenders
  - Products:
    - Utilities, Generators, Formatters
-  action links Hazards to Products
- Registry syncs hazard events locally & remotely
  - Products stored locally

Let's review some of the many important takeaways covered in this section.

One configuration workflow now applies to all hazards and products, once they are incorporated into Hazard Services, regardless of the legacy application they were previously managed through.

Thanks to its basis in very flexible python, almost ANYTHING can be edited, from the GUIs to the hazard metadata to the product formatting workflow, and more, greatly increasing the flexibility of this one workflow.

At the ground level of this workflow is a crucial distinction between hazard and product, reflected in the different components of Hazard Services and their configurations. Data related to hazards can be independently managed and shared, this task severed by critical, configurable components like hazard type definitions and categories, hazard-specific metadata, and recommenders. Meanwhile, a flexible framework for generating products handles simultaneous, multi-format generation, with configurable assembly and phrasing. Symbolically, these two functional groups are bridged when the user clicks "Preview," which transitions from the hazard world to product generation.

We also introduced the importance of the registry for synchronizing hazard data locally and remotely, which facilitates cooperation and planning from a hazard-centric

perspective.

In contrast, the fundamental product dictionaries and data are not synchronized, and remain local, although the resulting text products do get distributed exactly as before.



## Unified Configuration Takeaways 2

- Interoperability:
  - Hazard services *temporarily* cooperates with legacy applications
  - Overlapping legacy capabilities will be retired soon after
  - Parallel maintenance of changes necessary for functional side-by-side use
- Backup Operations:
  - Easy switching, and shared hazard data
  - Include Hazard Services directories in regular, up-to-date localization retrievals
- Support:
  - Focal Point Guide
  - Test-phase: [NWS.HazardServicesTeam.Staff@noaa.gov](mailto:NWS.HazardServicesTeam.Staff@noaa.gov), and Hazard\_services\_support NWS Chat room
  - Post-deployment: awips2dev listserv

In addition, we covered how, as Hazard Services is deployed, it will temporarily allow parallel use of the legacy software capabilities which it replaces, referred to as “interoperability.” During the limited overlap period where the legacy software is still supported, interoperability does require focal points to maintain legacy systems in parallel with Hazard Services for functional transition, when needed.

Backup operations are also supported in Hazard Services, with arguably some enhancement to this experience coming from the easy switching of sites, as well as the sharing of Hazard information through the registry. However localizations must, as always, be diligently updated for backup sites, now including Hazard Services localization directories. Retrieving these localizations remains, for now, a manual process.

The coming modules will dive deeper into many of the components discussed here, and other Hazard Services features. But for support, we’ve highlighted the availability of the Hazard Services Focal Point Guide for more in-depth reference on each component and their configuration. We’ve also introduced several channels for soliciting help, including the [NWS.HazardServicesTeam.Staff@noaa.gov](mailto:NWS.HazardServicesTeam.Staff@noaa.gov) email, and a dedicated NWS Chat room, both to be used during site testing, and, following deployment, the awips2dev listserv.

## Take the Quiz



You're almost finished!

Click "**Next**" to take the quiz.

[No audio on this slide]

What is used for configuring Hazard Services?

- Add and edit services
- Existing Hazard Services
- Stoppage
- Ready to go

## Unified Workflow Assessment

Quiz - 9 questions

Last Modified: Oct 09, 2018 at 07:13 PM

### PROPERTIES

On passing, 'Finish' button: [Goes to Next Slide](#)

On failing, 'Finish' button: [Goes to Next Slide](#)

Allow user to leave quiz: [After user has completed quiz](#)

User may view slides after quiz: [At any time](#)

Show in menu as: [Single item](#)



Edit in Quizmaker



Edit Properties